Available online at www.sciencedirect.com

ScienceDirect

Procedia Engineering 154 (2016) 1401 – 1405

Procedia Engineering

www.elsevier.com/locate/procedia

12th International Conference on Hydroinformatics, HIC 2016

Harmony search algorithm: A unique music-inspired algorithm

Joong Hoon Kim[a,*]

[a]*School of Civil, Environmental, and Architectural Engineering, Korea University, Anamdong 5ga 1, Seongbukgu, Seoul 136-713, Korea*

Abstract

Since the Harmony Search Algorithm (HSA) was first introduced in 2001, it has drawn a world-wide attention mainly because of its balanced combination of exploration and exploitation and ease of application. The HSA, inspired by musical performance process, consists of three operators: random search, harmony memory considering rule, and pitch adjusting rule. The ways of handling exploration and exploitation with the three operators make the HSA a unique metaheuristic algorithm. However, a series of papers was recently published by an author which insisted that the HSA is equivalent to an evolution strategy (ES). The ES, based on ideas of adaptation and evolution, consists of two operators: recombination and mutation operators. Except the similarity in generating a single new solution at each iteration which can replace the worst solution in the population, other components (e.g., their exploration and exploitation strategies and structure) are totally different between the HSA and ES. This paper is written to rebut and point out academic flaws in the papers.

Peer-review under responsibility of the organizing committee of HIC 2016

Keywords: Harmony search algorithm; Evolution strategy; Metaheuristic algorithm; Rebutal

1. Introduction

Despite the rebuttal [1] to the original claim that the harmony search algorithm (HSA) [2] is a special case of evolution strategies (ES): (μ+1)-ES, this argument has been restated in [3]. The Sudoku experiments in [4] were repeated but yielded inconsistent performance results [3]. In this paper, the author has focused on rebutting the

* Corresponding author. Tel.: +82-2-3290-3316; fax: +82-2-928-7656.
E-mail address: jaykim@korea.ac.kr

Peer-review under responsibility of the organizing committee of HIC 2016
doi:10.1016/j.proeng.2016.07.510

argument of the similarity of HSA and ES; it is expected that a response to the Sudoku issue will be made later by the original paper's author.

2. What are different between HS and ES?

First, the original rebuttal [1] is stated: Every meta-heuristic algorithm possesses similarity as well as uniqueness [1, 5]. Most metaheuristic algorithms are similar in their basic structure of exploration and exploitation operators. In general, the former involves the process of probing entirely new regions of a search space while the latter refers to the process of probing a limited but promising region of a design space. Every metaheuristic algorithm needs to have at least one operator that is responsible for each of the exploration and exploitation phases. In a genetic algorithm (GA), for instance, crossover and mutation operators are the exploitation and exploration operators, respectively.

Instead of checking whether two algorithms share certain operator(s) and/or structure, the uniqueness of an algorithm can be judged by the specific methods the algorithm uses for exploration and exploitation phases. The standard forms of HSA and (μ+1)-ES (which differ from that introduced in [3]) are similar in that a new solution is generated at each iteration, and can replace the worst solution in the harmony memory (HM) and population, respectively, if it is better than the worst solution. However, standard HSA consists of three operators (described below) while (μ+1)-ES consists of two operators (i.e., recombination and mutation operators). Standard HSA applies the harmony memory consideration (HMC) rule for both exploration and exploitation phases, random search (RS) for the exploration phase, and pitch adjusting (PA) rule for the exploitation phase. Indeed, the HMC rule acts as an exploration operator during the initial iterations because of large variations in the harmony memory (HM). However, as the iteration continues, this operator behaves as an exploitation agent for the HSA.

Therefore, prior to the argument [6], it had not been concluded that HSA is the same as (μ+1)-ES. This is because the exploration and exploitation strategies of these algorithms are different despite having a similar basic structure.

In addition, the two operations in (μ+1)-ES are mandatory while the frequency of each operation in HSA is controlled by the algorithm parameters (i.e., HMC rate and PA rate (PAR)). For example, the PA rule in HSA can occur only if the HMC rule has occurred, while the mutation operation in ES must be processed at every iteration. The PA rule addresses the local search, exploiting neighboring values of a decision variable. Unlike HSA, the Gaussian mutation operator is widely adopted in ES, and alters the decision variable by adding "a point symmetric perturbation" drawn from a multivariate normal distribution, with a zero mean and covariance matrix C, N(0, C). This is considered to be a very important parameter in ES [7].

As confirmed in [5], (μ+1)-ES has never been widely used. To the best of the author's knowledge, (μ+1)-ES with the two mutation operators described at lines 6 and 7 in Table 1 in [3] have never appeared in any earlier studies. Contrary to the declaration that "both (mutation) operators are very common mutation operators," (incidentally, the paper did not identify to which algorithm, ES or GA, the commonality applied), most references listed on page 4 in [3] following the statement do not prove that an ES uses such mutation operators. Surprisingly, mutation operators used in the cited references are also different from the PA rule used in the HSA. For instance, the mutation operator in [8] has equal probabilities for bigger and smaller decision variables, and the probability of a point perturbation being close to zero increases as the iteration increases, as shown below:

$$v_k^{'} = \begin{cases} v_k + \Delta(t, UB - v_k) & \text{if rand} \le 0.5 \\ v_k - \Delta(t, v_k - LB) & \text{if rand} > 0.5 \end{cases} \quad (1)$$

where $v_k^{'}$ is the kth component of a solution vector obtained after mutation, v_k is the component k of a solution vector after recombination, function $\Delta(t, y)$ returns a value in the range [0, y] such that the probability of $\Delta(t, y)$ being close to zero increases as t increases, and UB and LB denote the upper and lower bounds, respectively, for the kth component.

This is indubitably different from the PA rule of HSA that only considers the neighboring values of the decision variables as local search approaches, as shown below:

$$v_k' = \begin{cases} v_k \pm \Delta & \text{if rand} \le \text{PAR} \\ v_k & \text{if rand} > \text{PAR} \end{cases} \quad (2)$$

where Δ is the unit increment of the decision variable. Similar Gaussian mutations for GAs [9] were cited in support of the argument in [3].

3. Disguises

The author suspects an intention to create a (μ+1)-ES with the PA-similar mutation operator in order to support a predefined conclusion "HSA is equivalent to (μ+1)-ES" [3]. To further support this claim, a more detailed version of the pseudo code of ES was included in Table 1 [3]. The updated version added considerable detail on the recombination and mutation operators compared to the algorithm skeleton presented earlier [6]. It is worth noting that there was only one line "create a new solution using recombination and mutation operators" in the first version of the pseudo code.

Table 1. Pseudo code of (μ+1)-ES adopted from [3].

Algorithm 3: (μ + 1) evolution strategy with details about the recombination and mutation operators
1: initialize the population with μ randomly generated solutions
2: **repeat**
3: create a new solution in the following way
4: **for all** decision variables **do**
5: select a solution from the population uniformly at random and set the decision variable to the corresponding value of the selected solution
6: with probability p1 change the value slightly
7: with probability p2 change the value to a random one
8: **end for**
9: **if** the new solution is better than the worst solution in the population then
10: replace the worst solution by the new one
11: **end if**
12: until the maximum number of iterations has been reached
13: return the best solution in the harmony memory

However, the new pseudo code is still vague and unclear. For example, the pseudo code indicates that (μ+1)-ES changes the value of the decision variable "slightly" with probability p1 (see line 6 of Table 1). It is not clear what "slightly" indicates. This vague descriptor is also used in line 5 of the pseudo code of HSA (see Table 2), although standard HSA has unit increments or decrements according to the PA rule. There could have been better clarity when describing the PA rule. This could be an intention to dilute the exact mechanism of HSA to make it similar to (μ+1)-ES.

Table 2. Pseudo code of HSA adopted from [3].

Algorithm 1: The harmony search algorithm
1: initialize the harmony memory with HMS randomly generated solutions
2: **repeat**
3: create a new solution in the following way
4: **for all** decision variables **do**

5:	with probability HMCR use a value of one of the solutions in the harmony memory (selected uniformly at random) and additionally change this value slightly with probability PAR
6:	otherwise (with probability 1 - HMCR) use a random value for this decision variable
7:	**end for**
8:	**if** the new solution is better than the worst solution in the harmony memory then
9:	replace the worst solution by the new one
10:	**end if**
11:	**until** the maximum number of iterations has been reached
12:	**return** the best solution in the harmony memory

In addition, the updated pseudo code of (μ+1)-ES (Table 1) introduced more flaws along with the details. First, the author believes that the pseudo code was derived to support the predefined conclusion that "HSA is equivalent to (μ+1)-ES", because as mentioned before pure (μ+1)-ES does not have the mutation operator included in the pseudo code at lines 6 and 7 in Table 1. This fact can be easily verified by the references cited in [3] for the mutation operator of ES. For instance, other researchers [5, 7] have confirmed that the most general mutation operator adds a point symmetric perturbation to the result of recombination which is drawn from a multivariate normal distribution with a zero mean and covariance matrix C. Finally, "the harmony memory" in line 13 of the pseudo code for (μ+1)-ES (see Table 1) should be replaced with "the population."

The statement that HSA is a special case of ES was formally proven in [3] although the proposed (μ+1)-ES (Table 1) was already designed to mimic HSA by adopting unusual mutation operators that have never been used for ES. The mathematical proof of the theorem was used to give the false impression that the argument has a mathematical background, driven by the rules of a deductive system. That is, the mathematical statement was incorrectly established and proved to give the impression that the argument had some mathematical basis. Thus, it seems that the (μ+1)-ES was designed to be the same as HSA.

4. Final remarks

Finally, it seems that the meta-heuristic algorithms criticized in the Introduction section in [3] are not fully understood by the researcher, i.e., it is not really clear what the flow of water [10], leaps of frog [11], or a salmon run [12] have to do with optimization. The water flow algorithm was inspired by the hydrological cycle in meteorology and the erosion phenomenon in nature, and not just the flow of water. Many studies in the domain of geomorphology and hydrology have proven that optimality/equilibrium is involved in the natural hydrological and river network evolution process, such as a river network formation in order to minimize energy dissipation [13, 14] and so on. In addition, the main mechanism of the shuffled frog-leaping algorithm (SFLA) is not just the leaping of a frog but the population partitioned into several communities and the search information shuffling between them. Before criticizing and ridiculing other algorithms [3], the researcher should first carefully learn and study these algorithms. Why mislead readers by omitting very important information about the algorithms?

Acknowledgements

"This work was supported by a grant from The National Research Foundation (NRF) of Korea, funded by the Korean government (MSIP) (No. 2016R1A2A1A05005306)."

References

[1] Z. W. Geem, Research commentary: Survival of the fittest algorithm or the novelest algorithm? Int. J. Appl. Metaheuristic Comput., 1(4) (2010)

75–79.
[2] Z. W. Geem, J. H. Kim, G. V. Loganathan, A new heuristic optimization algorithm: harmony search. Simulation, 76(2) (2001) 60–68.
[3] D. Weyland, A critical analysis of the Harmony Search Algorithm - How not to solve Sudoku. Oper. Res. Perspect., 2 (2015) 97–105.
[4] Z. W. Geem, Harmony search algorithm for solving Sudoku. Knowledge-Based Intelligent Information and Engineering Systems, 371–378. Springer, 2007.
[5] T. Back, H. P. Schwefel, An overview of evolutionary algorithms for parameter optimization. Evolut. Comput., 1(1) (1993) 1–23.
[6] D. Weyland, A rigorous analysis of the Harmony Search Algorithm - How the research community can be misled by a "novel" methodology. International Journal of Applied Metaheuristic Computing, 1–2 (2010), pages 50–60.
[7] N. Hansen, D. Arnold, A. Auger. Evolution Strategies, Handbook of Computational Intelligence, 871–898, 2013.
[8] Z. Michalewicz, C. Z. Janikow, J. B. Krawczyk, A modified genetic algorithm for optimal control problems. Computers Math. Applic. 23(12) (1992), 83–94.
[9] D. B. Fogel and J. W. Atmar, Comparing genetic operators with Gaussian mutations in simulated evolutionary processes using linear systems. Biological Cybernetics, 63(2) (1990), 111–114.
[10] T. H. Tran and K. M. Ng, A water-flow algorithm for flexible flow shop scheduling with intermediate buffers. J. Schedul., 14(5) (2011) 483–500.
[11] M. Eusuff, K. Lansey, F. Pasha, Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization, Eng Optimiz, 38(2) (2006) 129–154.
[12] A. Mozaffari, A. Fathi, S. Behzadipour, The great salmon run: a novel bio-inspired algorithm for artificial system design and optimisation, International Journal of Bio-Inspired Computation 4(5) (2012) 286–301.
[13] I. Rodriguez-Iturbe, A. Rinaldo, R. Rigon, R. Bras, A. Marani, E. Ijjasz-Vasquez, Energy dissipation, runoff production, and the three-dimensional structure of river basins. Water Resour. Res., 28 (1992) 1095–1103.
[14] K. Paik and P. Kumar, Optimality approaches to describe characteristic fluvial patterns on landscapes, Philos. Trans. Royal Society B: Biol. Sci., 365(1545) (2010) 1387–1395.